```
#include "DSP281x_Device.h"
#include "example_nonBIOS.h"
#include "Init_Routines.h"
#include "x:\258 code\genmatlab.h"


// Function Calls

interrupt void TXIsr(void);     //McBSP Transmite interrupt service routine
interrupt void RXIsr(void);     //McBSP Recieve interrupt service routine
interrupt void Cap3ISR(void);   //Capture 3 interrupt service routine

void welcome(void);             //Display welcome screens
void disp_volume(char vol);     //Display voulume screen and initialize volume bar
void writenote(char);           //Write note to current LCD position

void home(void);                //Display home screen
void disp_notes(void);          //Display current notes
void settings(void);            //Display settings screen
void volume(void);              //Display volume adjustment screen
void credits(void);             //Display credits
void instrument(void);          //Display instrument selection screen
void probability(void);         //Display probability selection screen

void calcdata(void);            //Calculates the next 8 values to be sent to the McBSP
void setenv(void);              //Changes the envelope and harmonics data
void randomrhythm(void);
int randomnumber(void);         //Returns a random number in 1,2,..,n given a distru
                                //based on previous note


// Global Variables

int DataoutL[8]={0,0,0,0,0,0,0,0};      //Data to be sent to the McBSP, Left channel
int DataoutR[8]={0,0,0,0,0,0,0,0};      //Data to be sent to the McBSP, Right channel

long int y[15][3];                      //Recursive Oscilator Variables, 15 max harmonics
long int out;                           //Last stage befor dataout

unsigned char vol= 0xFF;                //Volume level
char vol2=20;                           //Volume bar level
char notechange = 0;                    //Notechange flag
char instsel = 0;                       //Instrument selection
char probsel = 0;                       //Probability delection
int se;                                 //Seed
char badnote = 0;                       //Bad note flag

int twonotesago = 0;                    //Two note ago
int lastnote = 0;                       //Previous note
char note=0;                            //Current note
```

```
Uint32 sample=0;                        //Current sample (resets every note, represents 8 points)
Uint32 length;                          //Cuurent note length in samples
Uint16 lastnotedur = 1;
Uint16 timesig = 1*44100/8;             //  4-4 time
Uint16 stepsize;
Uint16 extrasteps;
Uint16 step = 0;
Uint16 env[15][20];
int envharmonic;
int envstep;



void main(void)
{
    Uint16 i=0;

    //Call all initialization routines
    InitSysCtrl();
    InitGPIO();
    InitPieCtrl();
    InitInterrupts();
    InitMcBSP();

    // Section secureRamFuncs contains user defined code that runs from CSM secured RAM
    memcpy( &secureRamFuncs_runstart,
            &secureRamFuncs_loadstart,
            &secureRamFuncs_loadend - &secureRamFuncs_loadstart);

    /*** Initialize the FLASH ***/
    InitFlash();                        // Initialize the FLASH (FILE: SysCtrl.c)


    GpioDataRegs.GPADAT.all=0;          //Set Port a to 0x0000;

    wait(10);                           //Wait before initializing the LCD
    init_lcd();                         //Initialize the LCD

    setenv();                           //Setup the Oboe harmonics and envelope

    for(i=0 ; i < envharmonic ; i++)    //Initialize Difference Eq's
    {
        y[i][0]=0;
        y[i][1]=secval[i][note];
        y[i][2]=0;
    }

    welcome();                          //Display welcome screens
```

```
    //Increment se until button 2 is pressed

    while(GpioDataRegs.GPADAT.bit.GPIOA12 == 0)
    {
        se++;

        if(se>0xFE)
        {
            se=0;
        }
    }

    srand(se);                  //Set the rand seed

    note= randomnumber();       //Randomly generate the first note based on note 10 to note 12
    writenote(note);
    randomrhythm();
    length = NoteDur[lastnotedur];

    PieCtrlRegs.PIEIER6.bit.INTx6=1;    //Enable McBSP Interrupts and begin generating music


    while(1)
    {
        home();                 //Display the home screen
    }

}

#pragma CODE_SECTION(calcdata, "secureRamFuncs")
void calcdata(void)
{
    Uint16 i=1;
    Uint16 j=1;

    sample++;                               //Increment sample counter

    if(sample>length)                       //Transition to next note if sample > length
    {
        randomrhythm();

        length = NoteDur[lastnotedur];      //Generate next note length

        sample=0;                           //Reset sample
        step = 0;                           //Reset Step


        note=randomnumber();                //Generate next note

        for(i=0 ; i<envharmonic ; i++)      //ReInitialize Difference Eq's
```

```
        {
            y[i][0]=0;
            y[i][1]=secval[i][note];
            y[i][2]=0;
        }

        notechange = 1;                     //Set the note change flag
    }

    stepsize =  length/(envstep);
    extrasteps = length-(envstep*stepsize);
    if (sample >= (stepsize*(step+1)) && (step < envstep))
    {
        step++;
    }

    for( i=0 ; i<8 ; i++)                   //Generate next 8 samples
    {
        for(j=0 ; j<envharmonic ; j++)      //Update recursive oscilators
        {
            y[j][0] = (alpha[j][note]*y[j][1])/32767-y[j][2];
            y[j][2] = y[j][1];
            y[j][1] = y[j][0];
        }

        out = 0;                            //Reset Out
        for(j=0 ; j<envharmonic ; j++)      //Apply envelopes and sum oscilators
        {
            if (step >= envstep)
            {
            out=0;
            }
            else if (step==0)
            {
            out=out+(long int)(y[j][0]*(sample)*env[j][step])/(stepsize);
            }
            else
            {
            out=out+(long int)(y[j][0]*((sample-step*stepsize)*env[j][step]+((step+1)*stepsize-sample)*env[
            j][step-1]))/(stepsize);
            }
        }

        out=(out/255);

        if( vol != 0xFF )               //Adjust volume if not at 100%
        {
            out = (out*vol)>>8;
        }
```

```
        DataoutR[i]=out;                //Store data to be sent to McBSP
        DataoutL[i]=out;
        for (j=0; j<100;j++){}

    }
}

#pragma CODE_SECTION(randomnumber, "secureRamFuncs")
int randomnumber(void)
{
    Uint16 i,s,u;
    if (probsel == 0)
    {
        u = rand();

        i = 0;
        s = probtable[twonotesago][lastnote][i];

        if (s == 910)
        {
            badnote = 1;

        }

        while ((u > s))
        {
          i++;
          s=s+probtable[twonotesago][lastnote][i];
          if (i >= 36)
          {
              u = rand();
              i = 0;
              s = probtable[twonotesago][lastnote][i];
          }
        }
        twonotesago = lastnote;
        lastnote = i;
    }
    else
    {
        i = rand()%36;
    }
    return i;
}

void randomrhythm(void)  //Creates a Rhythm Phrase
{
    //global lastnotedur NoteDurOut timesig

//  Uint16 lastnotedurold = lastnotedur;
```

```
    Uint16 u = rand();
//  Uint16 durindex = 0;
//  Uint16 beat = 0;
    int i = 0;
    Uint16 s = rhythmtable[lastnotedur][i];

        while ((u > s))
        {
            i++;
            s=s+rhythmtable[lastnotedur][i];
        }


    lastnotedur = i;

}


// Interrupt Service Routines

interrupt void TXIsr(void)                  // McBSP transmite interrupt
{
    //Refill all 16 levels of FIFO
    McbspaRegs.DXR1.all=DataoutL[0];
    McbspaRegs.DXR1.all=DataoutR[0];
    McbspaRegs.DXR1.all=DataoutL[1];
    McbspaRegs.DXR1.all=DataoutR[1];
    McbspaRegs.DXR1.all=DataoutL[2];
    McbspaRegs.DXR1.all=DataoutR[2];
    McbspaRegs.DXR1.all=DataoutL[3];
    McbspaRegs.DXR1.all=DataoutR[3];
    McbspaRegs.DXR1.all=DataoutL[4];
    McbspaRegs.DXR1.all=DataoutR[4];
    McbspaRegs.DXR1.all=DataoutL[5];
    McbspaRegs.DXR1.all=DataoutR[5];
    McbspaRegs.DXR1.all=DataoutL[6];
    McbspaRegs.DXR1.all=DataoutR[6];
    McbspaRegs.DXR1.all=DataoutL[7];
    McbspaRegs.DXR1.all=DataoutR[7];

    calcdata();                             // Calculate next set of data

    McbspaRegs.MFFTX.bit.TXFFINT_CLEAR=1;   // Clear Interrupt flag
    PieCtrlRegs.PIEACK.all|=0x20;           // Issue PIE ack
}

interrupt void RXIsr(void)                  // Mcbsp FIFO recieve interrupt
{
    McbspaRegs.MFFRX.bit.RXFFOVF_CLEAR=1;   // Clear Overflow flag
    McbspaRegs.MFFRX.bit.RXFFINT_CLEAR=1;   // Clear Interrupt flag
```

```
    PieCtrlRegs.PIEACK.all|=0x20;           // Issue PIE ack
}


interrupt void Cap3ISR(void)                // Capture 3 interrupt
{
    EvaRegs.EVAIFRC.bit.CAP3INT = 1;        // Reset flag
    PieCtrlRegs.PIEACK.all|=0x20;           // Issue PIE ack
}

void setenv(void)
{
    Uint16 i,j;

    //0     Oboe
    //1     Trumpet
    //2     Trumbone
    //3     Flute
    //4     Marimba

    switch(instsel)
    {

        case 0:
            envharmonic = EnvHarmonicsoboe;
            envstep = EnvStepoboe;
            for (i=0; i < envharmonic; i++)
            {
                for (j=0; j < envstep; j++)
                {
                env[i][j] = Envflute[i][j];
                }
            }
            break;

        case 1:
            envharmonic = EnvHarmonicstrumpet;
            envstep = EnvSteptrumpet;
            for (i=0; i < envharmonic; i++)
            {
                for (j=0; j < envstep; j++)
                {
                env[i][j] = Envtrumpet[i][j];
                }
            }
            break;

        case 2:
            envharmonic = EnvHarmonicstrumbone;
            envstep = EnvSteptrumbone;
```

```
            for (i=0; i < envharmonic; i++)
            {
                for (j=0; j < envstep; j++)
                {
                env[i][j] = Envtrumbone[i][j];
                }
            }
            break;

        case 3:
            envharmonic = EnvHarmonicsflute;
            envstep = EnvStepflute;
            for (i=0; i < envharmonic; i++)
            {
                for (j=0; j < envstep; j++)
                {
                env[i][j] = Envflute[i][j];
                }
            }
            break;

        case 4:
            envharmonic = EnvHarmonicswack;
            envstep = EnvStepwack;
            for (i=0; i < envharmonic; i++)
            {
                for (j=0; j < envstep; j++)
                {
                env[i][j] = Envwack[i][j];
                }
            }
            break;
    }
}

void writenote(char note)
{
    char newn;
    if (note>=36)
    {
    write_lcd('o');
    write_lcd('v');
    }
    else if (note>=30)
    {
    newn = note -30;
    write_lcd('3');
    write_lcd(newn+0x30);
    write_lcd(' ');
    }
```

```
    else if (note>=20)
    {
    newn = note -20;
    write_lcd('2');
    write_lcd(newn+0x30);
    write_lcd(' ');
    }
    else if (note>=10)
    {
    newn = note -10;
    write_lcd('1');
    write_lcd(newn+0x30);
    write_lcd(' ');
    }
    else
    {
    newn = note;
    write_lcd(' ');
    write_lcd(newn+0x30);
    write_lcd(' ');
    }

}




// Display volume level screen
void disp_volume(char a)
{
    clear_lcd();

    move_lcd(7,0);

    write_lcd('V');
    write_lcd('o');
    write_lcd('l');
    write_lcd('u');
    write_lcd('m');
    write_lcd('e');
    move_lcd(0,1);
    write_lcd('0');
    write_lcd('%');
    move_lcd(16,1);
    write_lcd('1');
    write_lcd('0');
    write_lcd('0');
    write_lcd('%');
```

```
    draw_bar(2,a);
}


void welcome(void)
{
    Uint16 i=0;
    Uint16 j=0;

    char s1[4][20]=
    {
//   01234567890123456789
    "   Probabilistic    ",
    "  Music Generator   ",
    "                    ",
    "     Group 258      "
    };

    char s2[4][20]=
    {
//   01234567890123456789
    "  Generating  Seed  ",
    "                    ",
    "     Press > to     ",
    "      Continue      "
    };

    clear_lcd();

    for(j=0; j<4; j++)
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(s1[j][i]);
        }
    }

    wait(100);

    for(j=0; j<4; j++)
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(s2[j][i]);
        }
    }
```

```
}


void home(void)
{
    Uint16 i=0;
    Uint16 j=0;
    char sel=0;

    char home[4][20]=
    {
//   01234567890123456789
    "   Display Notes    ",     // Selection 0
    "   Settings         ",     // Selection 1
    "   Volume           ",     // Selection 2
    "   Credits          "      // Selection 3
    };

    clear_lcd();

    for(j=0; j<4; j++)                                      // Display Home Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(home[j][i]);
        }
    }

    move_lcd(1,0);                                          // Draw Cursor
    write_lcd(0x7E);

    wait(20);

    while(GpioDataRegs.GPADAT.bit.GPIOA12 == 0)
    {
        if(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)
        {
            move_lcd(1,sel);                                // Delete Cursor
            write_lcd(' ');

            sel++;                                          // Decrement selection

            if( sel == 4)
            {
                sel=0;
            }

            move_lcd(1,sel);                                //Move cursor
```

```
            write_lcd(0x7E);

            wait(1);                                        // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)     // Wait for release
            {
                asm("   nop");
            }
        }

        if(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)
        {

            move_lcd(1,sel);                                // Delete Cursor
            write_lcd(' ');

            if( sel == 0)
            {
                sel = 4;
            }

            sel--;                                          // Increment selection

            move_lcd(1,sel);                                // Move Cursor
            write_lcd(0x7E);

            wait(1);                                        // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)     // Wait for release
            {
                asm("   nop");
            }

        }

    }

    switch( sel )
    {

        case 0:
            disp_notes();
            break;

        case 1:
            settings();
            break;

        case 2:
            volume();
```

```
            break;

        default:
            credits();
            break;
    }

}



void disp_notes(void)
{
    Uint16 i=0;
    Uint16 j=0;

    clear_lcd();
    move_lcd(3,0);
    write_lcd('C');
    write_lcd('u');
    write_lcd('r');
    write_lcd('r');
    write_lcd('e');
    write_lcd('n');
    write_lcd('t');
    write_lcd(' ');
    write_lcd('N');
    write_lcd('o');
    write_lcd('t');
    write_lcd('e');
    write_lcd('s');

    move_lcd(1,1);

    //Display current notes until button 1 is pressed

    while( GpioDataRegs.GPADAT.bit.GPIOA11 == 0 )
    {
        if (notechange == 1)            //Display note if the note changes
        {
            notechange = 0;             //Reset the note change flag

            if( i == 18 )               //Move to line 1 if line 3 is full
            {
                i=0;
                move_lcd(0,1);
                for(j=0; j<19; j++)     //Clear line 1
                {
                    write_lcd(' ');
```

```
                }
                move_lcd(1,1);          //Move to begining of line 1

            }
            else if( i == 12)           //Move to line line 3 if line 2 is full
            {
                move_lcd(0,3);
                for(j=0; j<19; j++)     //Clear line 3
                {
                    write_lcd(' ');
                }
                move_lcd(1,3);          //Move to begining of line 3
            }
            else if( i == 6)            //Move to line line 2 if line 1 is full
            {
                move_lcd(0,2);          //Clear line 2
                for(j=0; j<19; j++)
                {
                    write_lcd(' ');
                }
                move_lcd(1,2);          //Move to begining of line 2

            }

            if (badnote == 1)           //Display "bad" if the note is a bad note
            {
            write_lcd('b');
            write_lcd('a');
            write_lcd('d');
            badnote = 0;                //Reset bad note flag
            }
            else
            {
            writenote(note);            //Else, display the current note
            }
            i++;                        //Increment note position
        }
    }

}


void settings(void)
{
    Uint16 i=0;
    Uint16 j=0;
    char sel=1;

    char settings[4][20]=
    {
```

```
//   01234567890123456789
    " Settings           ",     // Title
    "     Instrument     ",     // Selection 1
    "     Probability    ",     // Selection 2
    "                    "      // Selection 3
    };


    clear_lcd();

    for(j=0; j<4; j++)                                  // Display Settings Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(settings[j][i]);
        }
    }

    move_lcd(2,1);                                      // Draw Cursor
    write_lcd(0x7E);

    wait(20);

    while(GpioDataRegs.GPADAT.bit.GPIOA12 == 0)
    {
        if(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)
        {
            move_lcd(2,sel);                            // Delete Cursor
            write_lcd(' ');

            sel++;                                      // Decrement selection

            if( sel == 3)
            {
                sel=1;
            }

            move_lcd(2,sel);                            //Move cursor
            write_lcd(0x7E);

            wait(1);                                    // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)     // Wait for release
            {
                asm("   nop");
            }
        }

        if(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)
```

```
        {

            move_lcd(2,sel);                                    // Delete Cursor
            write_lcd(' ');

            if( sel == 1)
            {
                sel = 2;
            }

            sel--;                                              // Increment selection

            move_lcd(2,sel);                                    // Move Cursor
            write_lcd(0x7E);

            wait(1);                                            // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)         // Wait for release
            {
                asm("   nop");
            }

        }

    }

    switch( sel )
    {
        case 1:
            instrument();
            break;

        case 2:
            probability();
            break;

        default:
            break;
    }
}


void instrument(void)
{
    Uint16 i=0;
    Uint16 j=0;

    char harm[4][20]=
    {
//   01234567890123456789
```

```
    "Instrument Selection",     // Title
    "                    ",     // Selection 1
    "                    ",     // Selection 2
    "                    "      // Selection 3
    };

    char instlcd[5][20]=
    {
//   01234567890123456789
    "      Oboe          ",
    "      Trumpet       ",
    "      Trombone      ",
    "      Flute         ",
    "      Wack!!!       ",
    };



    clear_lcd();

    for(j=0; j<4; j++)                                  // Display Harmonics Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(harm[j][i]);
        }
    }


    move_lcd(0,2);
    for(i=0; i<20; i++)
    {
        write_lcd(instlcd[instsel][i]);
    }



    while(GpioDataRegs.GPADAT.bit.GPIOA11 == 0)
    {
        if(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)
        {

            if(instsel == 0)
            {
                instsel = 5;
            }

            instsel--;
```

```
            move_lcd(0,2);
            for(i=0; i<20; i++)
            {
                write_lcd(instlcd[instsel][i]);
            }

            setenv();

            wait(1);                                                // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)     // Wait for release
            {
                asm("   nop");
            }

        }

        if(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)
        {
            instsel++;

            if(instsel == 5)
            {
                instsel = 0;
            }

            move_lcd(0,2);
            for(i=0; i<20; i++)
            {
                write_lcd(instlcd[instsel][i]);
            }

            setenv();

            wait(1);                                                // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)     // Wait for release
            {
                asm("   nop");
            }

        }
    }
}

void probability(void)
{
    Uint16 i=0;
    Uint16 j=0;
```

```
    char prob[4][20]=
    {
//   01234567890123456789
    " Probability        ",     // Title
    "                    ",     //
    "                    ",     //
    "                    "      //
    };

    char select[2][20]=
    {
//   01234567890123456789
    "    Probabilistic   ",
    "    Random          "

    };


    clear_lcd();

    for(j=0; j<4; j++)                                  // Display selection
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(prob[j][i]);
        }
    }

    move_lcd(0,2);
    for(i=0; i<20; i++)
    {
        write_lcd(select[probsel][i]);
    }


    while(GpioDataRegs.GPADAT.bit.GPIOA11 == 0)
    {
        if( GpioDataRegs.GPADAT.bit.GPIOA13 == 1 || GpioDataRegs.GPADAT.bit.GPIOA14 == 1)
        {
            if( probsel == 0)
            {
                probsel = 1;
            }
            else
            {
                probsel = 0;
            }

            move_lcd(0,2);
```

```
            for(i=0; i<20; i++)
            {
                write_lcd(select[probsel][i]);
            }

            wait(40);                   //debounce
        }
    }
}


void volume(void)
{
    Uint16 i=20;
    Uint16 j=0;
    const char vol3[21]={0,3,5,8,14,23,27,32,38,44,52,61,71,84,98,115,135,158,185,217,255};

    char Warn[4][20]=
    {
//   01234567890123456789
    " WARNING !!         ",     // Selection 0
    " Reducing volume    ",     // Selection 1
    " reduces quality,   ",     // Selection 2
    " only use if needed."      // Selection 3
    };

    clear_lcd();

    for(j=0; j<4; j++)                                      // Display Warning Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(Warn[j][i]);
        }
    }

    wait(100);

    disp_volume(vol2);

    while(GpioDataRegs.GPADAT.bit.GPIOA11 == 0)
    {
        if(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)
        {
            if( vol2 > 0 )
            {
                vol2--;
                move_lcd(vol2,2);                           // Adjust Bar
                write_lcd(' ');
```

```
                vol = vol3[vol2];                           // Decrecrement volume
            }

            wait(1);                                        // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA14 == 1)     // Wait for release
            {
                asm("   nop");
            }
        }

        if(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)
        {
            if( vol2 < 20 )
            {
                move_lcd(vol2,2);                           // Adjust Bar
                write_lcd(0xFF);
                vol2++;
                vol = vol3[vol2];                           // Increcrement volume
            }

            wait(1);                                        // Debounce switch

            while(GpioDataRegs.GPADAT.bit.GPIOA13 == 1)     // Wait for release
            {
                asm("   nop");
            }
        }

    }

}

void credits(void)
{
    Uint16 i=0;
    Uint16 j=0;

    char Dave[4][20]=
    {
//   01234567890123456789
    "    David Loken     ",     // line 0
    "      Hardware      ",     // line 1
    "        and         ",     // line 2
    "       Coding       "      // line 3
    };

    char Chad[4][20]=
    {
//   01234567890123456789
```

```
    "    Chad Swenson    ",     // line 0
    "       Coding       ",     // line 1
    "         and        ",     // line 2
    "      Hardware      "      // line 3
    };

    char Bryce[4][20]=
    {
//   01234567890123456789
    "   Bryce Johnson    ",     // line 0
    "     Algorithms     ",     // line 1
    "        and         ",     // line 2
    "       Coding       "      // line 3
    };

    char DSP[4][20]=
    {
//   01234567890123456789
    "  DSP Scholar Team  ",     // line 0
    "      Funding       ",     // line 1
    "        and         ",     // line 2
    "      Hardware      "      // line 3
    };

    char Greg[4][20]=
    {
//   01234567890123456789
    "  Greg Middlestead  ",     // line 0
    "   Board Artwork    ",     // line 1
    "        and         ",     // line 2
    "      Hardware      "      // line 3
    };

    char Green[4][20]=
    {
//   01234567890123456789
    "  Dr. Roger Green   ",     // line 0
    "                    ",     // line 1
    "      Advisor       ",     // line 2
    "                    "      // line 3
    };


    clear_lcd();

    for(j=0; j<4; j++)                          // Display Dave's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
```

```
            write_lcd(Dave[j][i]);
        }
    }

    wait(100);

    for(j=0; j<4; j++)                                  // Display Chad's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(Chad[j][i]);
        }
    }

    wait(100);

    for(j=0; j<4; j++)                                  // Display Bryce's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(Bryce[j][i]);
        }
    }

    wait(100);

    for(j=0; j<4; j++)                                  // Display DSP's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(DSP[j][i]);
        }
    }

    wait(100);

    for(j=0; j<4; j++)                                  // Display Gregs's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(Greg[j][i]);
        }
    }

    wait(100);
```

```
    for(j=0; j<4; j++)                                  // Display Green's Screen
    {
        move_lcd(0,j);
        for(i=0; i<20; i++)
        {
            write_lcd(Green[j][i]);
        }
    }

    wait(100);

}
```